On the Termination of Flooding

Walter Hussak
Computer Science, Loughborough University, UK
W.Hussak@lboro.ac.uk

Amitabh Trehan[1]
Computer Science, Loughborough University, UK
www.amitabhtrehan.net
A.Trehan@lboro.ac.uk

Abstract

Flooding is among the simplest and most fundamental of all graph/network algorithms. Consider a (distributed network in the form of a) finite undirected graph G with a distinguished node v that begins flooding by sending copies of the same message to all its neighbours and the neighbours, in the next round, forward the message to all and only the neighbours they did not receive the message from in that round and so on. We assume that nodes do not keep a record of the flooding event, thus, raising the possibility that messages may circulate infinitely even on a finite graph. We call this history-less process *amnesiac flooding* (to distinguish from a classic distributed implementation of flooding that maintains a history of received messages to ensure a node never sends the same message again). Flooding will terminate when no node in G sends a message in a round, and, thus, subsequent rounds. As far as we know, the question of termination for amnesiac flooding has not been settled – rather, non-termination is implicitly assumed.

In this paper, we show that surprisingly synchronous amnesiac flooding always terminates on any arbitrary finite graph and derive exact termination times which differ sharply in bipartite and non-bipartite graphs. In particular, synchronous flooding terminates in e rounds, where e is the eccentricity of the source node, if and only if G is bipartite, and, otherwise, in j rounds where $e < j \leq e + d + 1$ and d is the diameter of G. Since e is bounded above by d, this implies termination times of at most d and of at most $2d + 1$ for bipartite and non-bipartite graphs respectively. This suggests that if communication/broadcast to all nodes is the motivation, the history-less amnesiac flooding is asymptotically time optimal and obviates the need for construction and maintenance of spanning structures like spanning trees. Moreover, the clear separation in the termination times of bipartite and non-bipartite graphs may suggest possible mechanisms for distributed discovery of the topology/distances in an arbitrary graph.

For comparison, we also show that, for asynchronous networks, however, an adversary can force the process to be non-terminating.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Graph algorithms analysis; Theory of computation → Distributed algorithms

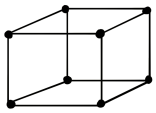
Keywords and phrases Flooding algorithm, Network algorithms, Distributed algorithms, Graph theory, Termination, Bipartiteness, Communication, Broadcast

Digital Object Identifier 10.4230/LIPIcs.STACS.2020.17

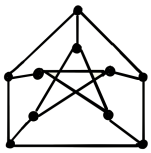
Related Version A brief announcement of this paper appeared at ACM PODC 2019 (https://doi.org/10.1145/3293611.3331586) and a related (Arxiv) version is available at https://arxiv.org/abs/1907.07078.

Funding *Amitabh Trehan*: This work was supported by the Engineering and Physical Sciences Research Council grant number EP/P021247/1.

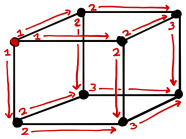
[1] corresponding author



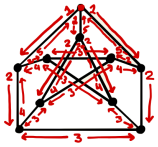
(a) The Hypercube graph (on 8 nodes).



(b) The Petersen Graph.



(c) Flooding on Hypercube.



(d) Flooding on the Petersen Graph.

Figure 1 Two well known graph topologies (Hypercube and the Petersen Graph) and execution of Amnesiac flooding (from the red coloured node) on them. The arrows point to direction of the transmission of the message with the label giving the round number. Double headed arrows indicate the message crossing over in both directions on the edge. The flooding on the hypercube terminates in only 3 = diameter rounds, whereas on the Petersen graph, it takes 5 = 2* diameter + 1 rounds.

Acknowledgements We would like to thank the anonymous reviewers for their comments, and Saket Saurabh, Jonas Lefèvre, Chhaya Trehan, Gary Bennett, Valerie King, Shay Kutten, Paul Spirakis, Abhinav Aggarwal for the useful discussions and insights and to all others in our network who attempted to solve this rather easy to state puzzle.

1 Introduction

Consider the two well known graphs in Figure 1; the hypercube (cube in 3 dimensions) graph and the Petersen graph. Now, consider distributed networks where nodes follow the following simple flooding process as a communication primitive: A single node (origin) with a message M begins the process by sending M to all its neighbours in the first round. These nodes, in the second round, in parallel forward M to all the other neighbours except the origin and so on. Nodes do this forwarding in a mechanical manner not retaining any memory, thus, forwarding M again if they receive it again. Possibly, this process can go on indefinitely. We call this process *Amnesiac Flooding (AF)* and define it more formally in later discussion. How does Amnesiac flooding behave if the topology of the network is a hypercube or a Petersen graph? What about other topologies?

Consider AF on the hypercube first (Figure 1(c)) - it is easy to see that it stops after 3 rounds when the node diagonally opposite the origin gets M from all of its neighbours simultaneously in round 3 and hence, cannot forward the message further. On the Petersen graph (Figure 1(d)), though the process terminates, it takes 5 rounds and stops at the origin itself. If we consider the termination times in terms of graph diameter, it takes diameter time on the hypergraph but much longer (2 times diameter plus 1) for the Petersen graph. Thus, the following question: *Will AF terminate on other network topologies, and if so, how long will it take? Why does the time differ markedly on the Hypercube and the Petersen graphs though they are of similar sizes (in fact, the Petersen graph has a smaller diameter)?*

Flooding is among the most basic of distributed graph/network algorithms. To quote Apnes [1]: *Flooding is about the simplest of all distributed algorithms. It is dumb and expensive, but easy to implement, and gives you both a broadcast mechanism and a way*

to build rooted spanning trees. At a high level, flooding can be simply described as: *In a network, a node begins flooding by sending a message to all its neighbours and subsequently, every node, in parallel, forwards the same message to all their neighbours.*

Flooding is the simplest strategy to achieve broadcast i.e. have a message reach every node in the network, in quick time. Often flooding is implemented with a flag that is set when the message is seen for the first time to ensure termination (see e.g. [2]) We are interested in the variant of flooding which does not explicitly use such a flag or keep a record of having seen the message before. The node selectively sends the message only to the complement of its neighbours from whom it has just received the message and subsequently forgets about that activity. The process terminates if there is no node that forwards M in a round (and, therefore, subsequent rounds). We call this amnesiac flooding (AF for short) to account for the very short term memory of the node. We analyse this very simple and theoretically interesting deterministic process on graphs and derive a rather unexpected and surprising result. We show that synchronous AF (i.e. in the synchronous message passing model where nodes send messages in parallel in synchronised rounds) terminates on every finite graph in time optimal $O(d)$ rounds, where d is the diameter of the graph. We also show that, at least in one asynchronous model, an adversary can force AF to be non-terminating.

Besides being theoretically interesting, our results also have practical implications. AF is a natural variant minimising memory overhead with nodes simply forwarding messages in a rather dumb manner. Note that if there were multiple messages being flooded in the network, the memory requirement of keeping the historical flags (for every message being flooded) could be significant, especially for low memory devices (e.g. sensor networks). Our results show that if the objective of the flooding is broadcasting, this overhead maybe unnecessary. Of course, a spanning substructure could be constructed from the initial regular flooding and used for subsequent broadcast (as is often done). However, spanning substructures can be difficult to maintain if the network is changing. This would not be required if AF was being used for communication.

We speculate (though we have not studied this in detail) that AF may correspond to certain natural and social phenomena to whose understanding our results may contribute. Consider the following possibly contrived example as a thought experiment: There is an aggressive social media user that forwards every message it receives to all its contacts but is polite enough to not forward to those who had just forwarded it the message. Naturally, such users lose track of the messages they have been forwarding. A natural question is that will a message cease getting circulated.

These need to be investigated further.

1.1 Model, Problem Definition and Results

Let $G(V, E)$ be an undirected graph (with n vertices and m edges) representing a network where the vertices represent the nodes of the network and edges represent the connections between the nodes. We consider the process in a synchronous message passing network: computation proceeds in synchronous rounds where each round consists of every node receiving messages from all its neighbours, doing local computation and sending messages to all (or some of) its neighbours. No messages are lost in transit. We consider only flooding from a single source for now.

▶ **Definition 1.** ***Synchronous Amnesiac Flooding (Synchronous AF):*** *A distinguished node, say ℓ, sends a message (say, M) to all its neighbours in round 1. In subsequent rounds, every node receiving M forwards a copy of M to every, and only those, nodes it did not receive the message from in the previous round. Algorithm 1.1 presents the algorithm formally.*

Algorithm 1.1 Synchronous Amnesiac Flooding: A message M from a source node s is "flooded" over graph G.

1: **procedure** *Flooding*(G, s) {Flooding over graph G from source node s}
2: Let $N(v) \leftarrow$ Neighbours of $v \in G$
3: Node s sends message M to all its neighbours in G {Round 1: s "floods" a message M}
4: **for** Rounds $i = 1, 2, \ldots$ **do**
5: **for** For all nodes v in parallel **do**
6: Let $I(v, M) \leftarrow$ set of neighbours of v that sent M to v in round $i-1$ $\{I(v, M) \subseteq N(v)\}$
7: Send M to $N(v) \setminus I(v, M)$ {Send to all neighbours except those who sent the message to v in the previous round}

Note that this is an "amnesiac" process i.e. nodes do not retain memory of having received or sent the message in the previous (but one) rounds. We say that flooding *terminates* when no message (i.e. a copy of M) is being sent over any edge in the network. We address the following questions:

For every finite graph G, beginning from any arbitrary vertex, will amnesiac flooding always terminate? If so, how many rounds does it take?

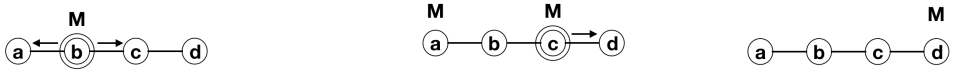
In Section 2, we answer the first part of the above question in the affirmative i.e. this flooding process will terminate for every G. For the second part of the question, in Section 3, we notice a sharp distinction between bipartite and non-bipartite graphs. Recall the standard definitions of *eccentricity* and *diameter*: eccentricity of a node is defined as the length of the longest of the shortest paths to other nodes in the graph, and diameter is the largest eccentricity of any node in the graph i.e. the longest distance between any two nodes in the graph. We show that flooding terminates in e rounds (i.e. at most d rounds), where e is the eccentricity of the source node and d the diameter of G, if and only if G is bipartite. Note that this is time optimal for broadcast. If the graph is non-bipartite, synchronous flooding takes longer: from a single source, flooding terminates in j rounds where $e < j \leq e + d + 1$.

Note that in this work, we only look at global termination i.e. the state when M stops circulating in the system. We do not discuss the related problem of individual nodes detecting that either global termination has happened or if they should stop participation in flooding. In some sense, this is even unnecessary since nodes do not need to maintain any additional state or history. There is no persistent overhead to keeping the simple amnesiac flooding process as a rule in the background.

1.1.1 Asynchronous Message Passing

For comparison, we also consider an asynchronous message passing model and show in Section 4 that an

adversary in this model can cause flooding to be non-terminating. We consider what we call as the *round-asynchronous model* where the computation still proceeds in global synchronous rounds but the adversary can decide the delay of message delivery on any link. The message cannot be lost and will be eventually delivered but the adversary can decide which round to deliver the message in. The adversary can decide on individual link delays for a round based on the state of the network for the present and previous rounds (i.e. node states, messages in transit and message history). Now, the flooding algorithm (*Asynchronous Amnesiac Flooding*) will exactly be same as Algorithm 1.1 except that the adversary decides which round a message transmitted on an edge reaches the other end. In Section 4, we show



(a) Round 1. **(b)** Round 2. **(c)** Round 3.

Figure 2 Amnesiac Flooding over a line network beginning with node b in 2 ($<$ diameter $= 3$) rounds. Circled nodes are sending M in that round.

(a) Round 1. **(b)** Round 2. **(c)** Round 3. **(d)** Round 4.

Figure 3 AF over a Triangle (Odd Cycle/Clique) network beginning with node b. Both node a and c send M to each other in round 2 and to b in round 3. Also, this is an odd (# nodes) cycle and termination takes $2d + 1$ time (d= diameter $= 1$).

that the adversary can force Asynchronous AF to be non-terminating by choosing link delays. Note that since the process is deterministic, the adversary can choose the link delays in advance (rather than needing to choose them adaptively).

We leave discussion of other asynchronous settings for future work.

1.2 Some illustrative examples

Figure 2 shows flooding over a line graph. The process begins with the node b and terminates at the ends of the graph and takes only 2 rounds, which is equal to the eccentricity of node b in the graph (which has diameter of 3). Note that a line is an example of a bipartite graph. The triangle graph is another interesting illustrative example (Figure 3) – here, termination takes 3 rounds, whereas, the diameter is only 1. Note that the triangle is also the smallest clique and the smallest non-trivial cycle with an odd number of nodes (an important topology for us). The even cycle is another interesting topology but here termination will happen in d rounds (as expected according to our bipartite graphs result). Of course, a graph can have far more complicated topology with cyclic and acyclic subgraphs.

1.3 Related work

A brief announcement of this work has appeared as [10]. The applications of flooding as a distributed algorithm are too numerous to be mentioned. It is one of the first algorithms to be introduced in distributed computing textbooks, often as the basic algorithm to solve leader election [12, 13] and set up graph substructures such as spanning trees [14, 2, 16, 19]. Flooding based algorithms (or flooding protocols) appear in areas ranging rom GPUs, High performance, shared memory and parallel computing to Mobile ad hoc networks(MANETs), Mesh Networks, Complex Networks etc [18]. In [17], Rahman et al show that flooding can even be adopted as a reliable and efficient routing scheme, comparable to sophisticated point-to-point forwarding schemes, in some ad-hoc wireless mobile network applications.

Termination is one of the most important properties a distributed algorithm requires. Since it is imperative to not have unnecessary messages circulating and clogging the network, explicit termination is desired and often enforced by using a flag to record if the node has

already participated in the flooding [1, 14, 2, 16, 19]. Otherwise, algorithms using broadcast for communication e.g. [9] (for high speed networks) use other explicit solutions to enforce termination. However, in some models such as population protocols, the low memory makes termination very difficult to achieve leading to research that tries to provide termination e.g. [15]. Our flooding algorithm has the advantage of being simple, using low memory, and being efficiently terminating as shown by our analysis. The idea of avoiding the most recently chosen node(s) has been used before in distributed protocols e.g. in social networks [6] and broadcasting [7] but we are not aware of this fundamental variant of flooding having been studied before. Lastly, processes such as random walks [4, 5, 8, 12, 13] and its deterministic variant Rotor-Router (or Propp) machine [11, 3] can be seen as restricted variants of flooding which possibly our work can provide some insight into.

2 Termination in a synchronous network

▶ **Definition 2.** *Let G be a graph. The round-sets $R_0, R_1, \ldots$ are defined as:*

R_0 *is the singleton containing an initial node,*
R_i *is the set of nodes which receive a message at round i* $(i \geq 1)$.

Clearly, if $R_j = \emptyset$ for some $j \geq 0$, then $R_i = \emptyset$ for all $i \geq j$. We shall refer to rounds R_i, where $R_i \neq \emptyset$, as *active* rounds.

▶ **Theorem 3.** *Any node $g \in G$ is contained in at most two distinct round-sets.*

Proof. Define $\mathcal{R}$ to be the set of finite sequences of consecutive round-sets of the form:

$$\underline{R} = R_s, \ldots, R_{s+d} \quad \textit{where } s \geq 0,\ d > 0, \textit{ and } R_s \cap R_{s+d} \neq \emptyset\ . \tag{1}$$

In (1), s is the *start-point* $s(\underline{R})$ and d is the *duration* $d(\underline{R})$ of $\underline{R}$. Note that, a node $g \in G$ belonging to R_s and R_{s+d} may also belong to other R_i in (1). If a node $g \in G$ occurs in three different round-sets R_{i_1}, R_{i_2} and R_{i_3}, then the duration between R_{i_1} and R_{i_2}, the duration between R_{i_2} and R_{i_3}, or the duration between R_{i_1} and R_{i_3} will be even. Consider the subset $\mathcal{R}^{EV}$ of $\mathcal{R}$ of sequences of the form (1) where d is even. To prove that no node is in three round-sets, it suffices to prove that $\mathcal{R}^{EV}$ is empty.

We assume that $\mathcal{R}^{EV}$ is non-empty and derive a contradiction.

Let $\mathcal{R}^{EV}_{\hat{d}}$ be the subset of $\mathcal{R}^{EV}$ comprising sequences of minimum (even) duration $\hat{d}$, i.e.

$$\mathcal{R}^{EV}_{\hat{d}} = \{\underline{R} \in \mathcal{R}^{EV} \mid \forall\, \underline{R}' \in \mathcal{R}^{EV}.\ d(\underline{R}') \geq d(\underline{R}) = \hat{d}\} \tag{2}$$

Clearly, if $\mathcal{R}^{EV}$ is non-empty then so is $\mathcal{R}^{EV}_{\hat{d}}$. Let $\underline{R}^* \in \mathcal{R}^{EV}_{\hat{d}}$ be the sequence with earliest start-point $\hat{s}$, i.e.

$$\underline{R}^* = R_{\hat{s}}, \ldots, R_{\hat{s}+\hat{d}} \tag{3}$$

where

$$\forall\, \underline{R}' \in \mathcal{R}^{EV}_{\hat{d}}\ .\ s(\underline{R}') \geq s(\underline{R}^*) = \hat{s} \tag{4}$$

By (1), there exists $g \in R_{\hat{s}} \cap R_{\hat{s}+\hat{d}}$. Choose node g' which sends a message to g in round $\hat{s} + \hat{d}$. As g' is a neighbour of g, either g' sends a message to g in round $\hat{s}$ or g sends a message to g' in round $\hat{s} + 1$. We show that each of these cases leads to a contradiction.

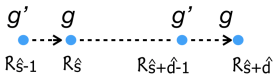


Figure 4 Node g' sends a message to node g in round $\hat{s}$: the first round of the minimum even length sequence (of length $\hat{d}$) in which g repeats.

Case (i): g' sends a message to g in round $\hat{s}$

Refer to Figure 4. In this case, there must be a round $\hat{s}-1$ which is either round 0 and g' is the initial node, or g' received a message in round $\hat{s}-1$. Thus, the sequence

$$\underline{R}^{*'} = R_{\hat{s}-1}, R_{\hat{s}}, \ldots, R_{\hat{s}+\hat{d}-1} \quad \textit{where } g' \in R_{\hat{s}-1} \cap R_{\hat{s}+\hat{d}-1} \tag{5}$$

has $d(\underline{R}^{*'}) = (\hat{s}+\hat{d}-1)-(\hat{s}-1) = \hat{d}$ which is even and so $\underline{R}^{*'} \in \mathcal{R}^{EV}_{\hat{d}}$. As $\underline{R}^{*'} \in \mathcal{R}^{EV}_{\hat{d}}$, by (4)

$$s(\underline{R}^{*'}) \geq s(\underline{R}^{*}) \tag{6}$$

But, from (5), $s(\underline{R}^{*'}) = \hat{s}-1$ and, from (4), $s(\underline{R}^{*}) = \hat{s}$. Thus, by (6),

$$\hat{s}-1 = s(\underline{R}^{*'}) \geq s(\underline{R}^{*}) = \hat{s}$$

which is a contradiction.

Case (ii): g sends a message to g' in round $\hat{s}+1$

Refer to Figure 5. By the definition of $\mathcal{R}^{EV}$, the smallest possible value of $\hat{d}$ is 2.

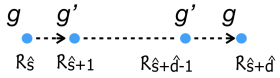


Figure 5 Node g sends a message to node g' in round $\hat{s}+1$: round $\hat{s}$ is the first round of the minimum even length sequence (of length $\hat{d}$) in which g repeats.

However, it is not possible to have $\hat{d}=2$ in this case as then

$$\underline{R}^{*} = R_{\hat{s}}, R_{\hat{s}+1}, R_{\hat{s}+2}$$

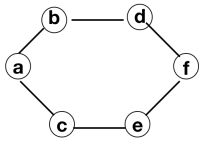
This would mean that g sends a message to g' in round $\hat{s}+1$. But, we chose g' to be such that g' sends a message to g in round $\hat{s}+\hat{d} = \hat{s}+2$. This cannot happen as g cannot send a message to g' and g' to g in consecutive rounds by the definition of rounds.

$$\underline{R}^{*} = R_{\hat{s}}, R_{\hat{s}+1}, \ldots, R_{\hat{s}+\hat{d}-1}, R_{\hat{s}+\hat{d}}$$

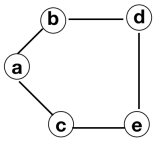
where $\hat{s}+1 < \hat{s}+\hat{d}-1$. Consider the sequence

$$\underline{R}^{*''} = R_{\hat{s}+1}, \ldots, R_{\hat{s}+\hat{d}-1} \tag{7}$$

As g' receives a message from g in round $\hat{s}+1$ and g' sends a message to g in round $\hat{s}+\hat{d}$, it is clear that $g' \in R_{\hat{s}+1} \cap R_{\hat{s}+\hat{d}-1}$. Thus, $\underline{R}^{*''} \in \mathcal{R}$. As $\hat{d}$ is even, so is



(a) Even Cycle Graph.



(b) Odd Cycle Graph.

Figure 6 Even cycle (6 nodes) and Odd cycle (5 nodes) graphs: Graphs show markedly different termination times. Consider AF from node b in both cases - In the 6-cycle it terminates in 3 rounds, but in the 5-cycle in 5 rounds.

$(\hat{s}+\hat{d}-1)-(\hat{s}+1)=\hat{d}-2$ and therefore $\underline{R}^{*''} \in \mathcal{R}^{EV}$. Now, $\underline{R}^{*} \in \mathcal{R}^{EV}_{\hat{d}}$ and so, as $\underline{R}^{*''} \in \mathcal{R}^{EV}$, we have, by (2),

$$d(\underline{R}^{*''}) \geq d(\underline{R}^{*}) \tag{8}$$

As $d(\underline{R}^{*''})=\hat{d}-2$ from (7) and $d(\underline{R}^{*})=\hat{d}$ from (3), we have, by (8),

$$\hat{d}-2=d(\underline{R}^{*''}) \geq d(\underline{R}^{*})=\hat{d}$$

This contradiction completes the proof. ◀

▶ **Definition 4.** *Given $g \in G$, we use a superscript 1 to indicate that g belongs to a round-set for the first time, and a superscript 2 to indicate that it belongs to a round-set for the second time, i.e.*

$$g^1 \in R_j$$

means that

$$g \in R_j \quad \textit{and} \quad g \notin R_i \quad \textit{for all } i \textit{ with } 0 \leq i < j.$$

and

$$g^2 \in R_j$$

means that

$$g \in R_j \quad \textit{and} \quad g \in R_i \quad \textit{for some } i \textit{ with } 0 \leq i < j.$$

Theorem 3 implies that $R_i = \emptyset$ for $i \geq 2n$, where n is the number of vertices of G, and therefore network flooding always terminates.

▶ **Corollary 5.** *Synchronous network flooding always terminates in fewer than $2n+1$ rounds.*

In the next section we give a greatly improved sharp upper bound for the number of rounds to termination, in terms of the eccentricity of the initial node and the diameter of G.

3 Time to termination

The question of termination of network flooding is non-trivial when cycles are present in G. The simple cases when G is an even cycle, as in Figure 6a and when G is an odd cycle, as in Figure 6b display quite different termination behaviours. The even cycle in Figure 6a

terminates after round e where e is the eccentricity of the initial node in G. On the other hand, flooding on the odd cycle in Figure 6b, returns a message to the initial node and terminates after round $2e+1$ resulting in a longer flooding process than the even cycle in Figure 6a despite having fewer nodes and a smaller value of e. In this section, we show that these observations can be largely generalized to arbitrary graphs. Specifically, we show that flooding on a graph G terminates after e rounds if and only if G is bipartite. If G is not bipartite, we show that flooding terminates after some round i where $e < i \leq e + d + 1$ and d is the diameter of G.

▶ **Definition 6.** *Let (G, E) be a graph with vertex set G and edge set E, and $g_0 \in G$ be an initial node. We will use the following definitions.*

(i) *For each $j \in \mathbb{N}$, the distance set D_j will denote the set of points which are a distance j from g_0. i.e.*

$$D_j = \{g \in G \ : \ d(g_0, g) = j\},$$

where d is the usual distance function in graph G.

(ii) *A node $g \in G$ is an equidistantly-connected node, abbreviated ec node, iff there there exists $g' \in G - \{g_0, g\}$ such that $d(g_0, g) = d(g_0, g')$ and $\{g, g'\} \in E$*

We have the following basic properties of distance sets D_j and *ec* nodes.

▶ **Lemma 7.** *Let G be a graph and $g_0 \in G$ be an initial node.*

(i) *For all $j \in \mathbb{N}$ and $i > j$, $D_j \subseteq R_j$ and $R_j \cap D_i = \emptyset$.*

(ii) *For all $j \in \mathbb{N}$, $g \in D_j$ and $g' \in D_{j+1}$ such that g and g' are neighbours, g sends a message to g' in round $j+1$, i.e. all nodes at a distance j from g_0 send to all their neighbours which are a distance $j+1$ in round $j+1$.*

(iii) *If $j \geq 1$ and $g \in D_j$ is an ec point, then $g^2 \in R_{j+1}$.*

Proof. For (i), clearly every node at a distance j from g_0 receives a message in round j and so $D_j \subseteq R_j$. Furthermore, every message received in round j will have travelled along j edges from g_0 and so could not have reached a node which is at a distance $i > j$ from g_0. Thus, $R_j \cap D_i = \emptyset$.

For (ii), we note that the only circumstance in which a node g in D_j ($\subseteq R_j$ by (i)) does not send to a neighbour g' in D_{j+1} in round $j+1$ is if g sent a message to g' in round j. This would need g to be in the round-set R_{j-1}, i.e. $g \in R_{j-1} \cap D_{j+1}$ which contradicts (i) which has $R_{j-1} \cap D_{j+1} = \emptyset$ as $j + 1 > j - 1$.

For (iii), if $j \geq 1$ and $g \in D_j$ is an *ec* point, then by Definition 6(ii) there is a point g' equidistant from the initial node g_0, i.e. $g' \in D_j$ such that g and g' are neighbours. By (i) of this lemma $D_j \subseteq R_j$, and so both g and g' receive messages in round j. Also by (i), neither sends a message in round j as $R_{j-1} \cap D_j = \emptyset$. Thus, g and g' send messages to each other in round $j+1$. As this will be the second time they receive messages we have that $g^2 \in R_{j+1}$. ◀

All nodes in a graph without *ec* nodes, belong to at most one round-set.

▶ **Lemma 8.** *Let G be a graph and let $g_0 \in G$ be an initial node. Then G has an ec node if and only if G has a node that is in two round-sets.*

Proof. Suppose that G has no *ec* nodes. Assume, on the contrary, that G has nodes that appear in two round-sets. Let R_j ($j \geq 1$) be the earliest round which contains a node g such that $g^2 \in R_j$ and $h \in R_{j-1}$ be a neighbour of g which sends to g in round j, so that

$h^1 \in R_{j-1}$. Then, $h \in D_i$ for some $i \geq 1$ and $h^1 \in R_i$ by Lemma i(i). Thus, $i = j - 1$ and so $g^2 \in R_{i+1}$. As g is a neighbour of h, $g \in D_i$, D_{i+1}, or D_{i-1}. If $g \in D_i$ then g and h are *ec* nodes contrary to our supposition that G has no *ec* nodes. If $g \in D_{i+1}$ then $g^1 \in R_{i+1}$ by Lemma i(i), which is contrary to the assertion that $g^2 \in R_j = R_{i+1}$. If $g \in D_{i-1}$ then $g \in R_{i-1}$, by Lemma i(i), and so $g^1 \in R_{i-1}$ as $g^2 \in R_{i+1}$. By Lemma ii(ii), g sends to h in round $i = j - 1$. This is contrary to h sending to g in round j. Thus, our assumption that G has nodes that appear in two round-sets is false.

Conversely, suppose that G has an *ec* node $g \in D_j$ where $j \geq 1$. Then $g^2 \in R_{j+1}$ by Lemma iii(iii). ◀

We note that bipartite graphs do not have any *ec* nodes.

▶ **Lemma 9.** *Let G be a graph and $g_0 \in G$ be an initial node. Then, G is bipartite iff it has no ec nodes.*

Proof. G is not bipartite iff there is a path from g_0 to an odd cycle in G. This is the case iff an edge of G connects two points equidistant from g_0, i.e G has an ec point. ◀

From Lemmas 8 and 9, we see that, in bipartite graphs, nodes only appear in one round-set. Thus, the time to termination can be determined by finding a bound on when each node belongs to a round-set.

▶ **Theorem 10.** *Let G be a graph and $g_0 \in G$ be an initial node with eccentricity e. Then, flooding will have terminated after round e if and only if G is bipartite.*

Proof.

G is bipartite	iff	G has no *ec* nodes	(by Lemma 9)
	iff	no node appears in 2 round-sets	(by Lemma 8)
	iff	R_e is the last non-empty round-set (as nodes a distance j from g_0 are only in R_j by Lemma i(i))	◀

To find the time to termination in general graphs we need to find a bound on when nodes can belong to a round-set for the second time. As nodes can only belong to at most two round-sets, by Theorem 3, this will give a bound for termination of flooding in general graphs. The following lemma relates the round-sets of second occurrences of neighbouring nodes.

▶ **Lemma 11.** *Let G be a graph and $g_0 \in G$ an initial node. If $h^2 \in R_j$ for some $j \in \mathbb{N}$, and if g is a neighbour of h, then*

$$g^2 \in R_{j-1} \text{ or } g^2 \in R_j \text{ or } g^2 \in R_{j+1}$$

Proof. Let i be the distance of h from g_0, i.e. $h \in D_i$. Then, as $h^2 \in R_j$, $j > i$ by Lemma i(i). As g is a neighbour of h, $g \in D_i$ or $g \in D_{i-1}$ or $g \in D_{i+1}$.

- *Case $g \in D_i$:* As $h, g \in D_i$ are neighbours they are both *ec* nodes. Thus, by Lemma iii(iii), $h^2 \in R_{i+1}$ and $g^2 \in R_{i+1}$. Therefore, $j = i + 1$ and $g^2 \in R_j$.
- *Case $g \in D_{i-1}$:* If $g \in R_j$ ($\neq R_{i-1}$ as $j > i$) then, as $g^1 \in D_{i-1} \subseteq R_{i-1}$ by Lemma i(i), it must be the case that $g^2 \in R_j$. If $g \notin R_j$ and $g \in R_{j-1}$ ($\neq R_{i-1}$ as $j > i$) then, as $g^1 \in R_{i-1}$ by Lemma i(i), it must be the case that $g^2 \in R_{j-1}$. If $g \notin R_j$ and $g \notin R_{j-1}$ then, as $h \in R_j$, h sends to g in round $j + 1$ and so $g \in R_{j+1}$ ($\neq R_{i-1}$ as $j > i$). As $g^1 \in R_{i-1}$, it must be the case that $g^2 \in R_{j+1}$.

- *Case $g \in D_{i+1}$, g does not send to h in round j:* In this case, as $h \in R_j$, h sends to g in round $j+1$. Thus, $g \in R_{j+1}$ ($\neq R_{i+1}$ as $j > i$) and therefore, as $g^1 \in D_{i+1} \subseteq R_{i+1}$ by Lemma i(i), it must be the case that $g^2 \in R_{j+1}$.
- *Case $g \in D_{i+1}$, g sends to h in round j:* In this case, $g \in R_{j-1}$. If $g^1 \in R_{j-1}$, then, by Lemma i(i) , $g^1 \in D_{i+1} \subseteq R_{i+1}$ and thus $j - 1 = i + 1$. Hence, by Lemma i(i), $h^1 \in D_i \subseteq R_i = R_{j-2}$. Also, $g \notin R_{j-3}$ as $g^1 \in R_{j-1}$.
  To summarize:

  $$g \notin R_{j-3}, \;\; h^1 \in R_{j-2}, \;\; g^1 \in R_{j-1}, \;\; h^2 \in R_j$$

  So, h sends to g in round $j-1$ and g sends to h in round j by the case assumption. This is a contradiction. Thus, $g^1 \notin R_{j-1}$ and, as $g \in R_{j-1}$, it follows that $g^2 \in R_{j-1}$.

This completes the proof. ◀

▶ **Theorem 12.** *Let G be a non-bipartite graph with diameter d and let $g_0 \in G$ be an initial node of eccentricity e. Then, flooding terminates after j rounds where j is in the range $e < j \leq e + d + 1$.*

Proof. If G is not bipartite it has an *ec* node g, by Lemma 9. By Lemma iii(iii), $g^2 \in R_k$ where $k = d(g_0, g) + 1$. Let h be an arbitrary node in G other than g. Then, there is a path

$$h_0 = g \longrightarrow h_1 \longrightarrow \ldots \longrightarrow h_l = h$$

where $l \leq d$. By repeated use of Lemma 11,

$$\begin{aligned}
&h_1^2 \in R_{j_1} \quad \text{where} \quad k - 1 \leq j_1 \leq k + 1,\\
&h_2^2 \in R_{j_2} \quad \text{where} \quad j_1 - 1 \leq j_2 \leq j_1 + 1,\\
&\ldots\\
&h_l^2 \in R_{j_l} \quad \text{where} \quad j_{l-1} - 1 \leq j_l \leq j_{l-1} + 1 \quad (l \geq 1).
\end{aligned}$$

Thus,

$$h_l^2 \in R_{j_l} \quad \textit{where} \quad k - l \leq j_l \leq k + l \tag{9}$$

Put $j = j_l$. From (19), as $k = d(g_o, g) + 1 \leq e + 1$ and as $l \leq d$,

$$h_l^2 \in R_j \quad \textit{where} \quad j \leq e + d + 1.$$

Thus, $j \leq e + d + 1$.
As G is not bipartite, $j > e$ by Theorem 10 and the proof is complete. ◀

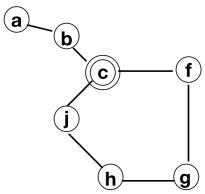
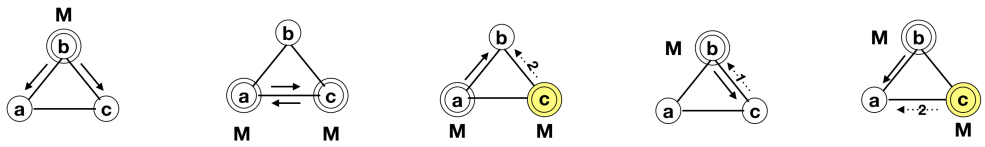


Figure 7 Flooding in the graph in the above figure starting from node c takes $e + d + 1$ rounds (the maximum as per our analysis), where e is eccentricity and d the diameter.

The upper bound in Theorem 12 is easily seen to be sharp - the flooding in the graph in Figure 7 starting from node c terminates after round $7 = 2 + 4 + 1 = e + d + 1$. Similar termination times hold for all nodes in the Petersen graph (Figure 1).



(a) Round 1 **(b)** Round 2 **(c)** Round 3 **(d)** Round 4 **(e)** Round 5

Figure 8 Asynchronous AF over a Triangle. Both node a and c send M to each other in round 2. In round 3, a sends M to b but the adversary makes c holds the message for one round (shaded node). In the next round, we have a round analogous to round 2 and so on.

4 Asynchronous Amnesiac Flooding

Non-termination in an adversarial asynchronous setting: Consider the *round-asynchronous* setting (as described in the model section). The scheduling adversary can choose the delay on every message edge i.e. which round to forward a message on.

An example suffices to prove non-termination. Consider round 3 in the triangle in Figure 8. The adversary delays M at node c but a continues and sends to b. In round 4, node b and c both send M so that the beginning of the next round is now identical to round 2 with nodes a and b interchanged. This process can now continue *ad infinitum* with the adversarial intervention.

5 Conclusion and Future Work

We studied a natural variant of the flooding algorithm where nodes do not retain any memory of the flooding beyond the previous round. We call this Amnesiac flooding (AF) and discussed the question of termination i.e. no copies of the initial message are being circulated anymore. We showed the surprising result that not only does this process terminate on all finite graphs but also accomplishes broadcast in almost optimal time and message overhead. There is a clear separation in complexity between bipartite and non-bipartite topologies. An interesting question is whether this separation can be exploited to devise distributed procedures to detect the topology of a graph given distance measures or vice versa. There is the question of multiple sources: what happens when multiple nodes start the flooding process with the same message M? We expect our method of proof can be extended to prove termination and obtain bounds in the case of multiple sources.

What about dynamic settings where nodes and edges change? It is easy to see that due to its simplicity, AF can be re-executed immediately after the graph has changed. However, what if the graph changes while messages are in circulation - under what conditions is termination/non-termination guaranteed?

Another important question is to look at flooding in asynchronous settings in more detail. We show one model where an adversary can force AF to be non-terminating. Since a completely asynchronous setting is event driven, this would also involve deciding what it means to receive messages simultaneously. Finally, one can see processes such as random walks, coalescing random walks and diffusion as probabilistic extremal variants of flooding. Are there any implications or connections of our result on these or intermediate probabilistic models? What about randomised variants of AF?

References

1 James Aspnes. Flooding, February 2019. URL: `http://www.cs.yale.edu/homes/aspnes/pinewiki/Flooding.html`.

2 Hagit Attiya and Jennifer Welch. *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. John Wiley & Sons, 2004.

3 JOSHUA N. COOPER and JOEL SPENCER. Simulating a random walk with constant error. *Combinatorics, Probability and Computing*, 15(6):815?822, 2006. `doi:10.1017/S0963548306007565`.

4 Atish Das Sarma, Danupon Nanongkai, and Gopal Pandurangan. Fast distributed random walks. In *PODC*, pages 161–170, 2009.

5 Atish Das Sarma, Danupon Nanongkai, Gopal Pandurangan, and Prasad Tetali. Efficient distributed random walks with applications. In *PODC*, 2010.

6 Benjamin Doerr, Mahmoud Fouz, and Tobias Friedrich. Social networks spread rumors in sublogarithmic time. *Electronic Notes in Discrete Mathematics*, 38:303–308, 2011. The Sixth European Conference on Combinatorics, Graph Theory and Applications, EuroComb 2011.

7 Robert Elsässer and Thomas Sauerwald. The power of memory in randomized broadcasting. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 218–227, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

8 C. Gkantsidis, M. Mihail, and A. Saberi. Random walks in peer-to-peer networks: Algorithms and evaluation. *Performance Evaluation*, 63(3):241–263, 2006.

9 Ajei S. Gopal, Inder S. Gopal, and Shay Kutten. Fast broadcast in high-speed networks. *IEEE/ACM Trans. Netw.*, 7(2):262–275, 1999. `doi:10.1109/90.769773`.

10 Walter Hussak and Amitabh Trehan. On termination of a flooding process. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019.*, pages 153–155. ACM, 2019. `doi:10.1145/3293611.3331586`.

11 Adrian Kosowski and Dominik Pajak. Does adding more agents make a difference? A case study of cover time for the rotor-router. *J. Comput. Syst. Sci.*, 106:80–93, 2019. `doi:10.1016/j.jcss.2019.07.001`.

12 Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. On the complexity of universal leader election. *J. ACM*, 62(1):7:1–7:27, 2015. `doi:10.1145/2699440`.

13 Shay Kutten, Gopal Pandurangan, David Peleg, Peter Robinson, and Amitabh Trehan. Sublinear bounds for randomized leader election. *Theoretical Computer Science*, 561(0):134–143, 2015. Special Issue on Distributed Computing and Networking. URL: `http://www.sciencedirect.com/science/article/pii/S0304397514001029`.

14 N. Lynch. *Distributed Algorithms*. Morgan Kaufmann Publishers, San Mateo, CA, 1996.

15 Othon Michail and Paul G. Spirakis. Terminating population protocols via some minimal global knowledge assumptions. *Journal of Parallel and Distributed Computing*, 81-82:1–10, 2015. `doi:10.1016/j.jpdc.2015.02.005`.

16 David Peleg. *Distributed Computing: A Locality Sensitive Approach*. SIAM, 2000.

17 A. Rahman, W. Olesinski, and P. Gburzynski. Controlled flooding in wireless ad-hoc networks. In *In Proceedings of IWWAN'04*, pages 73–78, 2004.

18 Andrew Tanenbaum. *Computer networks*. Pearson Prentice Hall, Boston, 2011.

19 Gerard Tel. *Introduction to distributed algorithms*. Cambridge University Press, New York, NY, USA, 1994.